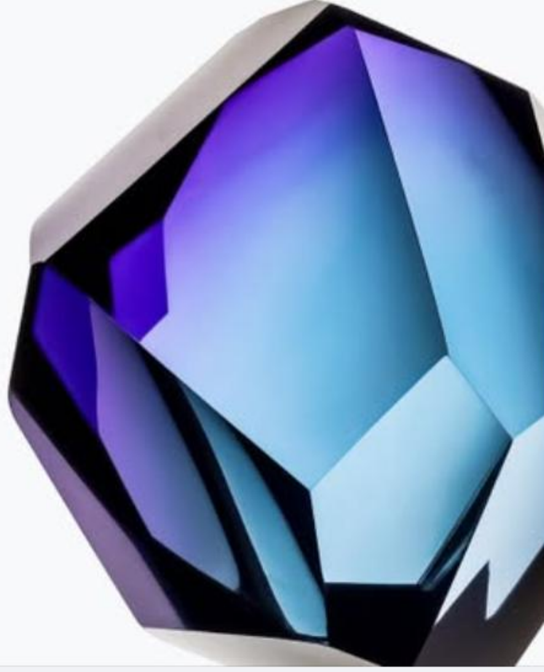
Prompt Engineering

Author: Lee Boonstra



Google

AI ki Duniya - اے آئی کی دنیا
Digital creator

Send message

گوگل کا پرامپٹ انجینئرنگ وائٹ پیپر

گوگل نے حال ہی میں ایک 69 صفحات پر مشتمل تفصیلی وائٹ پیپر (White Paper) جاری کیا ہے جو "Prompt Engineering" کے نام سے مشہور ہے۔ اس وائٹ پیپر کو مصنوعی ذہانت کے شوقین افراد، ڈویلپرز اور محققین کے لیے ایک مکمل رہنما (guide) کہا جا سکتا ہے۔ اس وائٹ پیپر کا بنیادی مقصد بڑے لینگویج ماڈلز (LLMs)

جیسے ChatGPT یا Gemini کے ساتھ بہتر طریقے سے بات چیت کرنے کا فن سکھانا ہے۔ یہ وائٹ پیپر واضح کرتا ہے کہ مصنوعی ذہانت کو بہتر طور پر کیسے سمجھایا جائے تاکہ وہ زیادہ مؤثر اور درست جواب دے سکے۔

یہ وائٹ پیپر صرف عام ہدایات پر مشتمل نہیں، بلکہ اس میں Prompt Engineering کے بنیادی تصورات، عملی تکنیکوں اور مؤثر اصولوں کو مرحلہ وار واضح کیا گیا ہے۔ یہ وضاحت کرتا ہے کہ Prompt Engineering صرف کمانڈز دینے تک محدود نہیں، بلکہ مصنوعی ذہانت کے ساتھ مؤثر رابطے قائم کرنے کا ایک مکمل فن ہے۔ آسان الفاظ میں، Prompt Engineering دراصل انسانوں اور مصنوعی ذہانت کے درمیان پل کی حیثیت رکھتا ہے۔ جتنی واضح اور مربوط آپ کی ہدایت یا سوال ہو گا، اتنا ہی بہتر اور درست جواب AI کی طرف سے ملے گا۔

یہ وائٹ پیپر متعدد Prompting تکنیکوں کو بھی تفصیلی طور پر واضح کرتا ہے۔ ان میں Zero-Shot Prompting شامل ہے، جس میں براہِ راست ہدایت دی جاتی ہے، One-Shot Prompting میں ایک مثال شامل ہوتی ہے، جبکہ Few-Shot Prompting میں متعدد مثالیں دی جاتی ہیں تاکہ AI زیادہ بہتر نتائج پیدا کر سکے۔ اس کے علاوہ پیچیدہ مسائل کو حل کرنے کے لیے Chain-of-Thought (CoT) تکنیک، جس میں AI قدم بہ قدم استدلال پیش کرتا ہے، اور ReAct (Reasoning + Action) تکنیک بھی شامل ہے جس میں AI صرف سوچتا ہی نہیں بلکہ عملی اقدامات بھی کرتا ہے۔ Code Prompting کے ذریعے مصنوعی ذہانت کو کوڈ لکھنے، سمجھنے، ترجمہ کرنے یا ڈی بگ کرنے پر بھی آمادہ کیا جا سکتا ہے۔

یہ وائٹ پیپر صرف تھیوری پر مبنی نہیں، بلکہ اس میں عملی مثالیں اور کیس اسٹڈیز بھی موجود ہیں جن سے واضح ہوتا ہے کہ کون سی تکنیک کس مسئلے کے لیے زیادہ موزوں ہے۔ مزید براں، گوگل نے اس وائٹ پیپر میں Prompt Engineering کے لیے بہترین عملی اصولوں (Best Practices) پر بھی روشنی ڈالی ہے۔ واضح، مختصر اور

مربوط ہدایات لکھنا، کام کی نوعیت کی وضاحت، پسِ منظر کی معلومات فراہم کرنا، مطلوبہ جواب کے فارمیٹ کو واضح کرنااور مستقل تجربات کے ذریعے اپنے پرامپٹس کو بہتر بناتے رہناان اصولوں میں شامل ہیں۔

یہ صرف ایک تکنیکی وائٹ پیپر نہیں بلکہ Prompt Engineering کے شعبے میں ایک اہم سنگ میل کی حیثیت رکھتاہے۔ مستقبل میں AI سے مؤثر رابطہ اور تعامل مزید اہم ہو جائے گا۔ یہی وجہ ہے کہ Prompt Engineering کو سمجھنااور سیکھنااب پہلے سے کہیں زیادہ اہمیت رکھتاہے۔

ہمارے فیس بک پیج پر آپ کو اس انتہائی اہم اور مفید وائٹ پیپر کا اردو زبان میں جامع خلاصہ چھ اقساط میں دستیاب ہو گا)اس سلسلے میں ہر قسط میں ان موضوعات کو مکمل وضاحت اور مثالوں کے ساتھ پیش کیا جائے گا۔ پرامپٹ انجینئرنگ کی بنیاد، LLM آؤٹ پٹ کنفیگریشن، ایڈوانس پرامپٹنگ ٹیکنیکس،:ایکشن پر مبنی پرامپٹنگ اور آٹومیٹک پرامپٹ انجینئرنگ، کوڈ پرامپٹنگ، موثر پرامپٹنگ کے بہترین طریقے اور تجاویز)۔ ہمارا مقصد ہے کہ اردو زبان بولنے اور سمجھنے والے افراد بھی اس انتہائی اہم میدان میں مہارت حاصل کر سکیں۔

یہ تحریر اے آئی کی دنیاکے فیس بک پیج پر پوسٹ کی گئی ہے۔

#AI #AIkiDuniya #google #Whitepaper #PromptEngineering

قسط نمبر 1: پرامپٹ انجینئرنگ کی بنیادیں(Basics of Prompt Engineering)

اس قسط میں ہم "Prompt Engineering" وائٹ پیپر کی روشنی میں پرامپٹ انجینئرنگ کے بنیادی اصول، اس کی تعریف، (Large Language Models (LLMs کے کام کرنے کا طریقہ کار، اور مختلف اقسام کی پرامپٹنگ(Zero-shot، Few-shot، System Prompting وغیرہ (کو تفصیل سے سمجھیں گے۔

پرامپٹ انجینئرنگ کیا ہے؟

پرامپٹ انجینئرنگ (Prompt Engineering) دراصل ایک ایسا عمل ہے جس میں خاص مقصد کے لیے مؤثر اور واضح ہدایات (prompts) لکھ کر بڑے لینگویج ماڈلز (LLMs) سے درست نتائج حاصل کیے جاتے ہیں۔ آسان الفاظ میں، پرامپٹ وہ متن (input) ہوتا ہے جس کے ذریعے لینگویج ماڈل اپنی ٹریننگ کی بنیاد پر مناسب جوابات یا نتائج کی پیشگوئی کرتا ہے۔ پرامپٹ انجینئرنگ کا مقصد ان prompts کو اس طرح ڈیزائن کرنا ہوتا ہے کہ ماڈل سے مطلوبہ نتائج ممکن حد تک واضح اور درست ہوں۔

لارج لینگویج ماڈلز (LLMs) Language Models کیسے کام کرتے ہیں؟

بڑے لینگویج ماڈلز بنیادی طور پر ایک prediction engine ہوتے ہیں۔ یہ ماڈلز ایک مسلسل متن (sequential text)بطور ان پٹ لیتے ہیں اور اس کے بعد اگلے ٹوکن (token) یعنی الفاظ، حروف یا علامات کی پیشگوئی کرتے ہیں۔ اس پیشگوئی کا عمل ماڈل کی سابقہ تربیت یافتہ معلومات کی بنیاد پر ہوتا ہے۔

پرامپٹ انجینئرنگ کے دوران، آپ LLM کو درست اور موزوں tokens کی ترتیب کو پیشگوئی کرنے کے لیے سیٹ اپ کرتے ہیں۔ اچھی پرامپٹ انجینئرنگ کے ذریعے آپ ماڈل کی صلاحیتوں کا بھرپور فائدہ اٹھاتے ہیں اور اپنی ضرورت کے عین مطابق نتائج حاصل کر سکتے ہیں۔

پرامپٹ کی مختلف اقسام(Prompting Techniques)

اس وائٹ پیپر میں درج ذیل بنیادی اقسام کا تفصیلی ذکر کیا گیا ہے:

1. جنرل پرامپٹنگ یا زیرو شاٹ(General prompting / Zero-shot)

زیرو شاٹ پرامپٹنگ سب سے بنیادی قسم ہے جس میں آپ ماڈل کو صرف ایک سوال، بیان یا ہدایت دیتے ہیں۔ اس میں کوئی اضافی مثال یا حوالہ شامل نہیں ہوتا۔

مثال:

vbnet

CopyEdit

" :Promptیہ فلمی تبصرہ مثبت، منفی یا معتدل ہے؟"

تبصرہ: "یہ فلم ایک منفرد شاہکار ہے جسے دیکھنا بہت خوشگوار تجربہ تھا۔"

ماڈل کی پیشگوئی: "مثبت"

یہاں ماڈل کو کوئی اضافی مثال دیے بغیر صرف ہدایت اور متن فراہم کیا گیا ہے۔

.2ون شاٹ اور فیو شاٹ پرامپٹنگ(One-shot & Few-shot)

• ون شاٹ پرامپٹنگ میں ماڈل کو ایک مثال فراہم کی جاتی ہے تاکہ وہ اس مثال سے رہنمائی لے کر کام مکمل کرے۔

• فیو شاٹ پرامپٹنگ میں ایک سے زیادہ (عام طور پر 3 سے 5) مثالیں دی جاتی ہیں، تاکہ ماڈل کو واضح طور پر سمجھ آئے کہ مطلوبہ آؤٹ پٹ کیسا ہونا چاہیے۔

مثال:(Few-shot)

vbnet

CopyEdit

Prompt:

فلمی تبصروں کو "مثبت"، "منفی" یا "معتدل" میں درجہ بندی کریں۔

مثال 1:

تبصرہ: "مجھے یہ فلم پسند نہیں آئی، یہ کافی بورنگ تھی۔"

Sentiment: "منفی"

مثال 2:

تبصرہ: "فلم اوسط درجے کی تھی، نہ بہت اچھی نہ بری۔"

Sentiment: "معتدل"

اب آپ درجہ بندی کریں:

تبصرہ: "یہ فلم ایک دلچسپ اور متاثر کن کہانی پر مبنی ہے۔"

ماڈل کی پیشگوئی: "مثبت"

یہاں ماڈل دی گئی مثالوں سے مدد لیتے ہوئے درست جواب تک پہنچتا ہے۔

3. سسٹم پرامپٹنگ (System Prompting)

سسٹم پرامپٹ وہ ہے جس میں ماڈل کو اضافی ہدایات یا وضاحت دی جاتی ہیں تاکہ وہ مخصوص نوعیت کا نتیجہ پیدا کرے۔

مثال:

makefile

CopyEdit

Prompt:

"فلمی تبصروں کو مثبت، منفی یا معتدل کے طور پر درجہ بندی کریں۔ جواب صرف بڑے حروف میں لکھیں۔"

تبصرہ: "فلم دیکھتے ہوئے بالکل مزا نہیں آیا۔"

Sentiment:

ماڈل کا جواب: "منفی"

اس طریقے سے ماڈل واضح اور مخصوص آؤٹ پٹ دیتا ہے۔

.4رول پرامپٹنگ (Role Prompting)

رول پرامپٹنگ میں ماڈل کو ایک خاص کردار یا شخصیت اختیار کرنے کو کہا جاتا ہے۔ مثلاً بطور ٹیچر، ٹریول گائیڈ، یا کامیڈین۔

مثال:

makefile

CopyEdit

Prompt:

آپ ایک مزاحیہ ٹریول گائیڈ ہیں۔ نیویارک میں تین بہترین جگہوں کی تجویز دیں۔

جواب:

.1 امپائر اسٹیٹ بلڈنگ پر جائیں اور شہر کا نظارہ کریں لیکن احتیاط کریں، کہیں کنگ کانگ نہ مل جائے!

.2موما (MoMA) میوزیم جائیں اور سوچیں کہ آپ کے بنائے ہوئے ڈرائنگز بھی "جدید آرٹ" ہو سکتی ہیں۔

.3ففتھ ایونیو پر شاپنگ کریں اور دیکھیں کہ مہنگی دکانوں کو دیکھ کر آپ کا بٹوہ کیسے روتا ہے!

یہاں ماڈل مزاحیہ انداز میں جواب فراہم کرتا ہے۔

.5 کونٹیکسچول پرامپٹنگ(Contextual Prompting)

اس تکنیک میں ماڈل کو مخصوص سیاق وسباق (context) فراہم کیا جاتا ہے تاکہ وہ موجودہ معلومات کے ساتھ متعلقہ جواب فراہم کرے۔

مثال:

markdown

CopyEdit

Prompt:

سیاق: آپ 1980 کی دہائی کے مشہور ویڈیو گیمز پر بلاگ لکھ رہے ہیں۔ اس موضوع پر 3 آرٹیکل تجاویز دیں۔

ماڈل کا جواب:

.1 آرکیڈ کیبنٹس کے ارتقا کی تاریخ

80 .2 کی دہائی کے مشہور ترین آرکیڈ گیمز

.3 پکسل آرٹ کا عروج اور اس کا دوبارہ ابھرنا

خلاصہ

اس قسط میں ہم نے پرامپٹ انجینئرنگ کی تعریف، بنیادی تصور، لینگویج ماڈلز کا کام کرنے کا طریقہ کار، اور مختلف پرامپٹ تکنیکوں کو تفصیل سے سمجھا ہے۔ پرامپٹ انجینئرنگ کی یہ بنیادی معلومات اس شعبے میں مہارت حاصل کرنے کے لیے نہایت ضروری ہیں۔

اگلی قسط میں ہم ماڈلز کی کنفیگریشن کے تفصیلی اصول (LLM Output Configuration) پر بات کریں گے، جس میں ٹوکن لمٹ، ٹیمپریچر، اور Sampling کنٹرولز شامل ہوں گے۔

یہ تحریر اے آئی کی دنیا کے فیس بک پیج پر پوسٹ کی گئی ہے۔

#AI #AIkiDuniya #google #Whitepaper #PromptEngineering

قسط نمبر 2 LLM : آؤٹ پٹ کنفیگریشن(Output Configuration)

اس قسط میں، ہم وائٹ پیپر "Prompt Engineering" کے مطابق Large Language Models (LLMs) کے آؤٹ پٹ کنفیگریشن کی تفصیلات بیان کریں گے۔ اس قسط میں خاص طور پر ٹوکن لمٹ (Token Limit)، Sampling کنٹرولز (Temperature، Top-K، اور Top-P) کی اہمیت اور ان کے مؤثر استعمال کے طریقوں کو تفصیلی طور پر سمجھیں گے۔

ٹوکن لمٹ (Token Limit) اور اس کی اہمیت

ٹوکن لمٹ دراصل وہ حد ہے جو LLM کے جواب میں شامل الفاظ یا ٹوکنز کی تعداد کو کنٹرول کرتی ہے۔ وائٹ پیپر کے مطابق یہ ایک اہم ترتیب ہے، کیونکہ زیادہ ٹوکنز کا مطلب ہے زیادہ کمپیوٹیشن، توانائی کا استعمال، اور اس کے نتیجے میں سست رفتار اور اضافی اخراجات۔

ٹوکن لمٹ کا استعمال مؤثر پرامپٹ انجینئرنگ کے لیے ناگزیر ہے کیونکہ یہ فیصلہ کرتا ہے کہ ماڈل جواب کب روکے گا۔ واضح رہے کہ ٹوکن لمٹ ماڈل کی تحریری قابلیت یا وضاحت پر اثرانداز نہیں ہوتی، بلکہ صرف جواب کی لمبائی کو محدود کرتی ہے۔ اس لیے اگر آپ مختصر جواب چاہتے ہیں، تو پرامپٹ کو بھی اس حساب سے ترتیب دینا ہوگا۔

خاص طور پر ReAct جیسی تکنیک کے لیے یہ ترتیب اہم ہے کیونکہ LLM اکثر ضروری جواب کے بعد غیر ضروری ٹوکنز بناتا رہتا ہے۔

Sampling کنٹرولز (Temperature، Top-K اور Top-P)

LLMs عام طور پر صرف ایک ٹوکن کی پیشن گوئی نہیں کرتے بلکہ وہ مختلف ٹوکنز کے امکانات (probabilities) کی پیشگوئی کرتے ہیں۔ ان ٹوکنز کے امکانات کو Sampling کے ذریعے چنا جاتا ہے۔ Temperature، Top-K اور Top-P ایسی ہی ترتیبات ہیں جو اس Sampling کے عمل کو کنٹرول کرتی ہیں۔

1. ٹمپریچر (Temperature)

ٹمپریچر ٹوکنز کی سلیکشن میں randomness (بے ترتیبیت) کو کنٹرول کرتا ہے۔ وائٹ پیپر کے مطابق، کم ٹمپریچر deterministic (واضح اور ٹھوس) جواب دیتا ہے جبکہ زیادہ ٹمپریچر غیر متوقع، تخلیقی اور متنوع نتائج دیتا ہے۔

- 0 ٹمپریچر مکمل deterministic ہوتا ہے اور ہمیشہ سب سے زیادہ امکان والے ٹوکن کو منتخب کرتا ہے۔

•زیادہ ٹمپریچر (مثلاً 1 یا اس سے زائد) بے ترتیبی کو بڑھاتا ہے، جس کے نتیجے میں جواب میں تخلیقیت بڑھ جاتی ہے لیکن جواب غیر متعلقہ بھی ہو سکتا ہے۔

2. ٹاپ کے سیمپلنگ۔Top-K Sampling

Top-K Sampling میں صرف ان K ٹوکنز کا انتخاب ہوتا ہے جن کے امکانات سب سے زیادہ ہوتے ہیں۔

•کم Top-K: جواب زیادہ factual اور مخصوص ہوتا ہے۔

•زیادہ Top-K: جواب متنوع اور تخلیقی ہوتا ہے۔

• Top-K = 1 greedy decoding کی مانند ہوتا ہے، یعنی صرف سب سے زیادہ امکان والا ٹوکن چنا جاتا ہے۔

3. ٹاپ پی۔Top-P (Nucleus Sampling)

Top-P بھی Sampling تکنیک ہے جس میں cumulative probability کی حد مقرر کی جاتی ہے۔ اس حد کے اندر آنے والے ٹوکنز ہی اگلے ٹوکن کے طور پر منتخب ہو سکتے ہیں۔

• Top-P قریب 0: صرف سب سے زیادہ امکان والا ٹوکن منتخب ہو گا۔

• Top-P قریب 1: تقریباً تمام ٹوکنز انتخاب کے لیے قابل غور ہوں گے۔

ان ترتیبات کو سمجھنے اور مؤثر طریقے سے استعمال کرنے کے طریقے

یہ ترتیبات ایک دوسرے کے ساتھ مل کر نتائج کو متاثر کرتی ہیں، لہٰذا انہیں سمجھنا اور درست طور پر سیٹ کرنا اہم ہے۔

وائٹ پیپر کے مطابق بہترین ترتیب کے لیے مشورہ:

• عمومی طور پر تخلیقی مگر مربوط جواب کے لیے:

o Temperature: 0.2

o Top-P: 0.95

o Top-K: 30

• زیادہ تخلیقی نتائج کے لیے:

o Temperature: 0.9

o Top-P: 0.99

o Top-K: 40

• کم تخلیقی اور زیادہ واضح نتائج کے لیے:

o Temperature: 0.1

o Top-P: 0.9

o Top-K: 20

• صرف ایک درست جواب کے لیے (مثلاً ریاضی کے مسائل میں):

o Temperature: 0

انتباہ:

یہ ترتیبات غلط سیٹ کرنے سے repetition loop یا filler words جیسے مسائل پیدا ہو سکتے ہیں۔ کم ٹمپریچر deterministic جوابوں کی وجہ سے looping پیدا کر سکتا ہے، جبکہ زیادہ ٹمپریچر random جوابوں کے ذریعے بھی looping کا سبب بن سکتا ہے۔ اس لیے ہمیشہ ان ترتیبات کے ساتھ احتیاط برتنا ضروری ہے۔

خلاصہ

اس قسط میں ہم نے ٹوکن لمٹ اور Sampling کنٹرولز (Temperature، Top-K، اور Top-P) کی تفصیلی اہمیت، ان کے کام کرنے کے طریقے اور مؤثر استعمال کے لیے وائٹ پیپر "Prompt Engineering" کی روشنی میں واضح ہدایات بیان کیں۔ ان ترتیبات کو سمجھنا اور صحیح طریقے سے استعمال کرنا ایک مؤثر پرامپٹ انجینئر بننے کے لیے ضروری ہے۔

اگلی قسط میں ہم "ایڈوانس پرامپٹنگ تکنیکس (Advanced Prompting Techniques) "کو تفصیل سے زیر بحث لائیں گے۔

یہ تحریر اے آئی کی دنیا کے فیس بک پیج پر پوسٹ کی گئی ہے۔

#AI #AIkiDuniya #google #Whitepaper #PromptEngineering

قسط نمبر 3: ایڈوانس پرامپٹنگ تکنیکس (Advanced Prompting Techniques)

اس قسط میں ہم وائٹ پیپر "Prompt Engineering" کے مطابق ایڈوانس پرامپٹنگ تکنیکس بشمول Chain of Thought (CoT)، Self-consistency اور (😭) Tree of Thoughts کو تفصیلی طور پر بیان کریں

گے۔ یہ تکنیکس خاص طور پر پیچیدہ مسائل کو حل کرنے اور LLM کے جوابات میں زیادہ گہرائی اور وضاحت پیدا کرنے کے لیے استعمال کی جاتی ہیں۔

1۔ چین آف تھاٹ (CoT) Chain of Thought پرامپٹنگ کیا ہے اور کیسے مدد کرتی ہے؟

(CoT) Chain of Thought ایک ایسی پرامپٹنگ تکنیک ہے جس میں LLM کو جواب دینے سے پہلے مرحلہ وار سوچنے اور اپنے جواب کی منطقی وجہ (reasoning steps) فراہم کرنے پر آمادہ کیا جاتا ہے۔ اس کا مقصد LLM کی reasoning کی صلاحیتوں کو بہتر بنانا ہے۔

چین آف تھاٹ Chain of Thought کی اہم خصوصیات:

- Low-effort and High-impact: آسان لیکن انتہائی مؤثر۔
- Reasoning Transparency: اس تکنیک سے ماڈل کی reasoning واضح ہو جاتی ہے، جس سے آپ ماڈل کی سوچ کے طریقہ کار کو سمجھ سکتے ہیں۔
- Robustness: ماڈل کے ورژن تبدیل ہونے پر بھی نتائج مستحکم رہتے ہیں۔

مثال (وائٹ پیپر سے ماخوذ):

سادہ Prompt بغیر CoT کے):

pgsql

CopyEdit

When I was 3 years old, my partner was 3 times my age. Now, I am 20 years old. How old is my partner?

غلط جواب: 63 سال

CoT Prompt (Step-by-step سوچنے پر مجبور کرنے والی):

pgsql

CopyEdit

When I was 3 years old, my partner was 3 times my age. Now, I am 20 years old. How old is my partner? Let's think step by step.

صحیح جواب:

- جب میری عمر 3 سال تھی تو میرے پارٹنر کی عمر 9 سال تھی۔
- اب میری عمر 20 سال ہے، یعنی 17 سال کا اضافہ ہوا۔
- میرے پارٹنر کی موجودہ عمر ہو گی: 9 + 17 = 26 سال۔

2۔ Self-consistency اور اس کی عملی اہمیت

Self-consistency تکنیک Chain of Thought کی توسیع شدہ صورت ہے، جو LLM سے متعدد reasoning paths حاصل کرکے ان میں سے زیادہ consistent جواب کو منتخب کرنے پر مبنی ہے۔ یہ تکنیک accuracy اور coherence کو بڑھاتی ہے۔

سیلف Self-consistency کا عمل:(Step by Step)

.1 ایک سوال کو بار بار LLM کو پیش کرنا (high temperature کے ساتھ)۔

.2 ہر دفعہ ماڈل مختلف reasoning فراہم کرتا ہے۔

.3 آخر میں، سب سے زیادہ ظاہر ہونے والے جواب کو منتخب کیا جاتا ہے۔

مثال:(Email Classification)

• Prompt:

vbnet

CopyEdit

(EMAIL: ایک پیچیدہ ای میل کا متن دیا گیا)

Classify the above email as IMPORTANT or NOT IMPORTANT. Let's think step by step and explain why.

• مختلف جواب:

o کچھ دفعہ IMPORTANT

o کچھ دفعہ NOT IMPORTANT

• حتمی جواب: زیادہ تر صورتوں میں جو جواب آیا (IMPORTANT) وہ منتخب ہو گا۔

3۔ (😭) Tree of Thoughts تکنیک کی وضاحت اور مثالیں

(😭) Tree of Thoughtsتکنیک CoT اور Self-consistency سے بھی زیادہ ایڈوانس ہے۔ اس تکنیک میں LLM ایک ہی وقت میں reasoning کے کئی راستے explore کرتا ہے۔ اس تکنیک میں سوچوں کا ایک درخت (tree) تشکیل دیا جاتا ہے جس میں مختلف سوچوں کے راستے branch کی شکل میں نکلتے ہیں اور مختلف ممکنہ جوابات پر غور کیا جاتا ہے۔

ٹری آف تھاٹ۔ Tree of Thoughtsکے فوائد:

• پیچیدہ مسائل کے لیے مؤثر۔

• متعدد reasoning paths کی وجہ سے مسائل کا زیادہ تفصیلی حل ممکن ہوتا ہے۔

• یہ ماڈل کو زیادہ جامع exploration کی اجازت دیتا ہے، جس سے جواب کی accuracy میں اضافہ ہوتا ہے۔

مثال (تصویری خاکہ):

• ا۔ Chain of Thought صرف ایک سیدھی لائن میں جواب تلاش کرتا ہے۔

• ب۔ Tree of Thoughts مختلف راستوں پر بیک وقت غور کرتے ہوئے ایک درخت کی شکل میں جواب کو تلاش کرتا ہے (وائٹ پیپر کے صفحہ 36 پر تصویری مثال موجود ہے جو واضح کرتی ہے کہ 😭کس طرح سے مختلف pathsکو explore کرتا ہے)۔

خلاصہ

اس قسط میں ہم نے وائٹ پیپر "Prompt Engineering" کی روشنی میں ایڈوانس پرامپٹنگ تکنیکس، جیسے (CoT) Chain of Thought، Self-consistencyاور (😭) Tree of Thoughts کو سمجھا۔ ہم نے

ان تکنیکس کی وضاحت مثالوں کے ساتھ کی، جن سے ان تکنیکس کے استعمال کا عملی اور واضح اندازہ ہوا۔ یہ تکنیکس LLM کے ذریعے پیچیدہ اور گہری reasoning والے مسائل کو حل کرنے کے لیے انتہائی مفید ہیں۔

اگلی قسط میں ہم "ایکشن پر مبنی پرامپٹنگ اور آٹومیٹک پرامپٹ انجینئرنگ (Action-based Prompting & Automatic Prompt Engineering)" کو تفصیل سے زیرِ بحث لائیں گے۔

یہ تحریر اے آئی کی دنیا کے فیس بک پیج پر پوسٹ کی گئی ہے۔

#AI #AIkiDuniya #google #Whitepaper #PromptEngineering

قسط نمبر 4: ایکشن پر مبنی پرامپٹنگ اور آٹومیٹک پرامپٹ انجینئرنگ

اس قسط میں ہم "Prompt Engineering" وائٹ پیپر کی روشنی میں دو انتہائی اہم ایڈوانس تکنیکوں یعنی ReAct (Reason and Act) اور Automatic Prompt Engineering (APE) کو تفصیلی طور پر سمجھیں گے۔ یہ دونوں تکنیکس LLM کی صلاحیتوں کو مزید وسیع کرنے میں معاون ہوتی ہیں اور پیچیدہ مسائل کو بہتر طور پر حل کرنے میں مدد دیتی ہیں۔

1۔ ReAct (Reason and Act) تکنیک کیا ہے اور عملی مثال

ReAct (Reason & Act) ایک ایسا طریقہ کار ہے جو LLMs کو قدرتی زبان میں استدلال (reasoning) کے ساتھ ساتھ ایکشن (action) کرنے کی بھی اجازت دیتا ہے۔ اس تکنیک میں LLM مسئلے پر سوچ بچار (reason) کے بعد عملی اقدامات (actions) بھی کرتا ہے، جیسے بیرونی ٹولز (مثلاً سرچ انجن) کے ذریعے معلومات حاصل کرنا۔

ReAct کا عملی طریقہ کار:

ReAct دراصل انسانی سوچ اور عمل کرنے کے طریقہ کو نقل کرتا ہے:

• ماڈل پہلے مسئلے پر غور وفکر کرتا ہے (reasoning) ۔

• پھر وہ ایک عملی منصوبہ بناتا ہے (plan of action) ۔

• اس کے بعد وہ منصوبے پر عمل کرتے ہوئے معلومات جمع کرتا ہے۔

• جمع شدہ معلومات کو دوبارہ غور وفکر میں استعمال کر کے نئے اقدامات کا تعین کرتا ہے۔

• اس لوپ کو بار بار دہراتے ہوئے، مسئلہ حل ہونے تک جاری رکھتا ہے۔

وائٹ پیپر سے ماخوذ عملی مثال (صفحہ 39):

LLM کو سوال دیا گیا:

"Metallica بینڈ کے ممبران کے کتنے بچے ہیں؟"

ReAct تکنیک کا استعمال کرتے ہوئے ماڈل نے مندرجہ ذیل مراحل طے کیے:

1. بینڈ ممبران کی تعداد (4) معلوم کی۔

2. ہر ممبر کے بچوں کی تعداد الگ الگ سرچ کی:

o James Hetfield کے 3 بچے

o Lars Ulrich کے 3 بچے

o Kirk Hammett کے 2 بچے

o Robert Trujillo کے 2 بچے

.3 آخر میں تمام اعداد جمع کر کے نتیجہ دیا: کل 10 بچے۔

یہ مثال ReAct کی طاقت کو ظاہر کرتی ہے کہ کس طرح LLM صرف سوچنے کی بجائے عملی اقدامات بھی کر سکتا ہے۔

2۔ آٹو میٹک پر امپٹ انجینئرنگ (Automatic Prompt Engineering – APE) کا طریقہ کار

آٹو میٹک پر امپٹ انجینئرنگ (APE) ایک ایسی تکنیک ہے جس میں LLM خود مزید بہتر پر امپٹس تیار کرتا ہے۔ اس تکنیک کے ذریعے پر امپٹ تیار کرنے کا عمل مکمل یا جزوی طور پر خود کار ہو جاتا ہے۔ یہ تکنیک خاص طور پر اس لیے مفید ہے کہ یہ انسانی مداخلت کم کرتے ہوئے ماڈل کی صلاحیتوں میں اضافہ کرتی ہے۔

APE تکنیک کا عملی طریقہ:

• ماڈل کو بنیادی پر امپٹ دیا جاتا ہے۔

• LLM مختلف اقسام کے نئے پر امپٹس تیار کرتا ہے۔

• ان تیار کردہ پر امپٹس کو معیار کی بنیاد پر جانچا جاتا ہے۔

• بہترین پر امپٹ کا انتخاب کر کے اس پر مزید بہتری کی جاتی ہے۔

وائٹ پیپر سے ماخوذ عملی مثال (صفحہ 41):

فرض کریں آپ ایک ٹی شرٹ ویب شاپ کے چیٹ بوٹ کو ٹرین کرنا چاہتے ہیں۔ آپ کو صارفین کے ٹی شرٹ آرڈر کرنے کے مختلف طریقوں کے لیے پرامپٹس تیار کرنے ہیں:

بنیادی پرامپٹ:

arduino

CopyEdit

"One Metallica t-shirt size S" کو بیان کرنے کے لیے مختلف طریقے لکھیں۔

ماڈل کی جانب سے تیار کردہ کچھ پرامپٹس:

- I'd like to purchase a Metallica t-shirt in size small.
- Can I order a small-sized Metallica t-shirt?
- One Metallica shirt, size small, please.

ان میں سے بہترین پرامپٹ منتخب کرنے کے بعد، اسے مزید بہتر بنا کر حتمی استعمال کے لیے تیار کیا جاتا ہے۔

ان دونوں تکنیکوں کے عملی استعمال کی اہمیت اور فوائد

ReAct کے فوائد:

- پیچیدہ مسائل کو بہتر طریقے سے حل کرنے میں معاون۔
- عملی اقدامات کرنے کی صلاحیت کی وجہ سے LLM زیادہ عملی اور متعلقہ نتائج فراہم کرتا ہے۔
- یہ تکنیک LLM کو حقیقی دنیا کے استعمال کے لیے مزید مؤثر بناتی ہے۔

Automatic Prompt Engineering کے فوائد:

• انسانی کوشش کم ہوتی ہے۔

• پرامپٹ بنانے کا عمل تیز، خودکار اور مؤثر ہو جاتا ہے۔

• ماڈل کی صلاحیتوں کو زیادہ سے زیادہ استعمال کرنے میں مددگار ثابت ہوتا ہے۔

خلاصہ

اس قسط میں ہم نے "Prompt Engineering" وائٹ پیپر کی روشنی میں ReAct اور Automatic Prompt Engineering (APE) کی تکنیکس کو تفصیلی طور پر سمجھا۔ ReAct تکنیک ماڈل کو عملی اقدامات کی صلاحیت فراہم کرتی ہے جبکہ APE خودکار طریقے سے بہتر پرامپٹس بنانے میں مدد دیتی ہے۔ ان تکنیکوں کے ذریعے آپ LLM کی صلاحیتوں کو مزید بڑھا سکتے ہیں اور پیچیدہ مسائل کو حل کرنے میں آسانی حاصل کر سکتے ہیں۔

اگلی قسط میں ہم "کوڈ پرامپٹنگ "(Code Prompting) کے بارے میں تفصیلی گفتگو کریں گے۔

یہ تحریر اے آئی کی دنیا کے فیس بک پیج پر پوسٹ کی گئی ہے۔

#AI #AIkiDuniya #google #Whitepaper #PromptEngineering

قسط نمبر 5: کوڈ پرامپٹنگ (Code Prompting)

اس بورنگ (کچھ لوگوں کے لیے) قسط میں ہم وائٹ پیپر "Prompt Engineering" کی روشنی میں کوڈ پرامپٹنگ کے بارے میں تفصیل سے بات کریں گے۔ ہم دیکھیں گے کہ کس طرح LLMs کوڈ لکھنے، سمجھنے، ترجمہ

کرنے اور ڈی بگ کرنے کے لیے استعمال کیے جاسکتے ہیں۔ اس کے علاوہ ہم مؤثر کوڈ پرامپٹنگ کے اصول اور عملی مظاہروں کو بھی واضح مثالوں کے ساتھ سمجھیں گے۔

1۔ کوڈ لکھنے کے لیے پرامپٹس کیسے لکھیں؟

وائٹ پیپر کے مطابق، LLMs جیسے Gemini، نہ صرف عام متن کے لیے بلکہ مختلف پروگرامنگ زبانوں میں کوڈ لکھنے کے لیے بھی استعمال کیے جاسکتے ہیں۔ آپ کو بس واضح اور جامع پرامپٹ لکھنا ہوگا جس میں مطلوبہ پروگرامنگ زبان اور مقصد کی وضاحت کی گئی ہو۔

عملی مثال (Bash اسکرپٹ لکھنے کی مثال، صفحہ 43):

پرامپٹ:

nginx

CopyEdit

Bash میں کوڈ لکھیں جو صارف سے فولڈر کا نام پوچھے، اس فولڈر کے اندر موجود تمام فائلز کے ناموں کے شروع میں "draft" لگادے۔

جواب (ماڈل سے حاصل شدہ Bash کوڈ):

bash

CopyEdit

#!/bin/bash

```
echo "Enter the folder name: "
read folder_name
if [ ! -d "$folder_name" ]; then
echo "Folder does not exist."
exit 1
fi
files=("$folder_name"/*)
for file in "${files[@]}"; do
new_file_name="draft_$(basename "$file")"
mv "$file" "$new_file_name"
done
echo "Files renamed successfully."
```

اس مثال میں ماڈل نے واضح اور جامع Bash اسکرپٹ فراہم کیا جو مکمل طور پر کام کرنے والا ہے۔

2۔ کوڈ سمجھنے اور وضاحت کرنے کے لیے پرامپٹس

LLMs موجودہ کوڈ کو پڑھ کر اس کی تفصیلی وضاحت فراہم کرنے کے لیے بھی استعمال کیے جا سکتے ہیں۔ یہ خصوصیت خاص طور پر اس وقت مفید ہے جب آپ ٹیم کے دیگر ممبران کے لکھے ہوئے کوڈ کو سمجھنا چاہتے ہوں۔

عملی مثال (صفحہ 45):

پرامپٹ (Bash کوڈ کی وضاحت):

bash

CopyEdit

درج ذیل Bash کوڈ کو واضح کریں:

```
#!/bin/bash
echo "Enter the folder name: "
read folder_name
if [ ! -d "$folder_name" ]; then
echo "Folder does not exist."
exit 1
fi
files=( "$folder_name"/* )
for file in "${files[@]}"; do
new_file_name="draft_$(basename "$file")"
```

```
mv "$file" "$new_file_name"
done
echo "Files renamed successfully."
```

وضاحت (LLM سے حاصل کردہ جواب):

• فولڈر نام پوچھنا

• فولڈر کے موجود ہونے کی تصدیق

• فولڈر میں تمام فائلز کو لسٹ کرنا

• ہر فائل کے نام کے آغاز میں "draft_" لگا کر فائل کا نام تبدیل کرنا

• آخر میں کامیابی کا پیغام دکھانا۔

3۔ کوڈ ترجمہ کرنے کے لیے پرامپٹس

LLMs مختلف پروگرامنگ زبانوں کے درمیان کوڈ کا ترجمہ بھی کر سکتے ہیں۔ وائٹ پیپر میں Bash سے Python میں کوڈ ترجمہ کرنے کی واضح مثال موجود ہے۔

عملی مثال (صفحہ 47):

پرامپٹ:

CopyEdit

درج ذیل Bash کوڈ کو Python میں ترجمہ کریں:

Bashکوڈ:

```bash
bash
CopyEdit
#!/bin/bash
echo "Enter the folder name: "
read folder_name
if [ ! -d "$folder_name" ]; then
echo "Folder does not exist."
exit 1
fi
files=( "$folder_name"/* )
for file in "${files[@]}"; do
new_file_name="draft_$(basename "$file")"
mv "$file" "$new_file_name"
done
```

echo "Files renamed successfully."

Python میں ترجمہ شدہ کوڈ (LLM کا جواب):

python

CopyEdit

```
import os
import shutil
folder_name = input("Enter the folder name: ")
if not os.path.isdir(folder_name):
print("Folder does not exist.")
exit(1)
files = os.listdir(folder_name)
for file in files:
new_file_name = f"draft_{file}"
shutil.move(os.path.join(folder_name, file), os.path.join(folder_name, new_file_name))
print("Files renamed successfully.")
```

یہاں LLM نے واضح اور درست Python کوڈ فراہم کیا ہے۔

4۔ کوڈ ڈی بگ کرنے کے لیے پرامپٹس

LLMs خراب یا غلط کوڈ کو ٹھیک کرنے اور اس کی اصلاح کرنے میں بھی مؤثر ہیں۔

عملی مثال (صفحہ 49):

غلط Python کوڈ:

python

CopyEdit

```
folder_name = input("Enter folder name: ")
prefix = input("Enter prefix: ")
text = toUpperCase(prefix) # غلط فنکشن کا نام
```

پرامپٹ:

CopyEdit

مندرجہ بالا کوڈ میں موجود غلطی کو تلاش کریں اور بہتر کوڈ فراہم کریں۔

LLM سے موصول درست کوڈ اور وضاحت:

python

CopyEdit

```
text = prefix.upper()
```

وضاحت:

toUpperCase فنکشن Python میں موجود نہیں، اس کی جگہ upper() استعمال کریں۔

5۔ مؤثر کوڈ پرامپٹنگ کے اصول اور رہنمائی

وائٹ پیپر سے ماخوذ کچھ اہم اصول:

- واضح اور مکمل ہدایات: ہمیشہ واضح اور مکمل پرامپٹس لکھیں۔
- مخصوص پروگرامنگ زبان کی وضاحت: ہمیشہ پروگرامنگ زبان کو واضح طور پر بیان کریں۔
- کوڈ ٹیسٹنگ: LLM کے فراہم کردہ کوڈ کو عملی طور پر ٹیسٹ کریں۔
- اضافی معلومات: جیسے تبصرے یا واضح ڈیٹا ٹائپس شامل کریں۔

خلاصہ

اس قسط میں ہم نے وائٹ پیپر "Prompt Engineering" کی مدد سے LLM کے ذریعے کوڈ لکھنے، سمجھنے، ترجمہ کرنے، اور ڈی بگ کرنے کی تکنیکس کو واضح مثالوں کے ساتھ سمجھا۔ یہ واضح ہو گیا کہ کوڈ پرامپٹنگ کے ذریعے ہم ترقیاتی عمل کو آسان، تیز اور مؤثر بنا سکتے ہیں۔

اگلی اور آخری قسط میں ہم "بہترین طریقے اور تجاویز (Best Practices and Tips)" کے بارے میں تفصیلی گفتگو کریں گے۔

قسط نمبر 6: (آخری قسط) بہترین طریقے اور تجاویز (Best Practices and Tips)

اس قسط میں ہم "Prompt Engineering" وائٹ پیپر کی روشنی میں مؤثر پرامپٹس لکھنے کے لیے بہترین طریقے، اہم تجاویز اور عملی اصولوں کو تفصیلی طور پر بیان کریں گے۔ یہ تجاویز آپ کو بہترین نتائج حاصل کرنے میں مدد فراہم کریں گی اور آپ کی پرامپٹ انجینئرنگ کی صلاحیتوں کو مزید بہتر بنائیں گی۔

1۔ مؤثر پرامپٹس لکھنے کی اہم تجاویز

وائٹ پیپر کے مطابق مؤثر پرامپٹس لکھنے کے لیے مندرجہ ذیل بنیادی نکات کو ہمیشہ ذہن میں رکھیں:

) Simplicity سادگی(

• ہمیشہ مختصر، واضح اور سادہ زبان استعمال کریں۔

• غیر ضروری یا مبہم معلومات سے گریز کریں۔

• پیچیدہ الفاظ اور جملوں کے بجائے آسان اور واضح ہدایات دیں۔

مثال:

• غلط: میں نیویارک میں ہوں اور میرے ساتھ 3 سال کے دو بچے ہیں۔ ہمیں کہاں جانا چاہیے؟

• صحیح: بطور ٹریول گائیڈ 3 سال کے بچوں کے ساتھ نیویارک میں گھومنے کی بہترین جگہیں بتائیں۔

وضاحت (Specificity)

• جوابات کی مطلوبہ شکل اور طرز کو واضح طور پر بیان کریں۔

• عمومی اور غیر واضح سوالات سے بچیں، اور خاص طور پر ہدف کو واضح کریں۔

مثال:

• غلط: ویڈیو گیمز کے بارے میں بلاگ پوسٹ لکھیں۔

• صحیح: 5 مشہور ترین ویڈیو گیم کنسولز پر 3 پیراگراف پر مشتمل ایک معلوماتی اور دلچسپ بلاگ پوسٹ لکھیں۔

مثالوں کا استعمال (Provide Examples)

• ہمیشہ پرامپٹ میں مثالیں شامل کریں (One-shot یا Few-shot)۔

• مثالوں سے ماڈل کو مطلوبہ نتیجہ کا واضح اندازہ ہوتا ہے جس سے اس کے نتائج مزید درست ہوتے ہیں۔

2۔ مختلف فارمیٹس (مثلاً JSON) کا مؤثر استعمال

JSON یا دیگر اسٹرکچرڈ فارمیٹس کے استعمال سے LLM کی پیداوار کو بہتر بنایا جا سکتا ہے۔ JSON کے فوائد درج ذیل ہیں:

• ہمیشہ یکساں فارمیٹ میں جواب ملتا ہے۔

• واضح ڈیٹا اسٹرکچر کی وجہ سے نتائج میں غلطیوں (hallucinations) کا امکان کم ہو جاتا ہے۔

• ڈیٹا واضح طور پر منظم اور قابل فہم ہوتا ہے۔

JSON استعمال کرنے کی مثال:

json

CopyEdit

```
{
"movie_reviews": [
{
"sentiment": "POSITIVE",
"movie": "Her"
}
]
}
```

JSON استعمال کی احتیاط:

- JSON کا فارمیٹ زیادہ ٹوکنز لیتا ہے جس کی وجہ سے کمپیوٹیشنل لاگت بڑھ سکتی ہے۔
- JSON کے جوابات کا ٹوکن لمٹ سے باہر کٹ جانا بھی ممکن ہوتا ہے، جسے درست کرنے کے لیے خصوصی JSON repair ٹولز استعمال کیے جاسکتے ہیں۔

3۔ ہدایات (Instructions) بمقابلہ پابندیاں (Constraints)

- Instructions (ہدایات): واضح طور پر بتائیں کہ آپ کیا چاہتے ہیں۔
- Constraints (پابندیاں): مخصوص پابندیوں کو واضح کرتی ہیں کہ ماڈل کو کیا نہیں کرنا چاہیے۔

عام طور پر واضح ہدایات زیادہ مؤثر ہوتی ہیں۔

مثال:

• ہدایت (صحیح): 5 مقبول ویڈیو گیمز پر صرف کنسول، کمپنی کا نام، سال، اور کل فروخت کا ذکر کرتے ہوئے ایک پیراگراف لکھیں۔

• پابندی (کم مؤثر): ویڈیو گیمز کے نام مت لکھیں۔

4۔ ماڈل کی اپ ڈیٹس کے ساتھ پرامپٹس کی ایڈجسٹمنٹ اور وائٹ پیپر کی اہمیت

LLM کے ماڈلز مسلسل اپ ڈیٹ ہوتے رہتے ہیں۔ ان اپ ڈیٹس کے مطابق پرامپٹس کی ایڈجسٹمنٹ ضروری ہے۔ یہ یقینی بنانے کے لیے کہ آپ ہمیشہ بہترین نتائج حاصل کریں، مندرجہ ذیل اقدامات کریں:

• باقاعدگی سے ماڈل اپ ڈیٹس کی نگرانی کریں۔

• نئے ماڈل پر پرانے پرامپٹس کو دوبارہ ٹیسٹ کریں۔

• پرامپٹس کی وائٹ پیپر بندی (Documentation) کریں تاکہ مستقبل میں ان میں آسانی سے تبدیلیاں کی جا سکیں۔

وائٹ پیپر کی مثال (Table 21، صفحہ 66):

تفصیل مثال

نام movie_review_v1

مقصد فلمی تبصروں کی درجہ بندی کرنا

ماڈلgemini-pro

ٹمپریچر 0.1

ٹوکن لمٹ256

Top-K 30

Top-P 0.9

پرامپٹ تبصرہ کو مثبت، منفی یا معتدل میں درجہ بندی کریں۔

جواب مثبت

5۔ دیگر مفید تجاویز(Additional Tips)

• ٹوکن لمٹ کنٹرول: جواب کی لمبائی کے لیے ٹوکن لمٹ واضح کریں۔

• پرامپٹ میں متغیرات :(Variables) پرامپٹس کو لچکدار بنانے کے لیے متغیرات استعمال کریں۔

• پرامپٹ فارمیٹس کے ساتھ تجربات: مختلف انداز اور اسلوب کی آزمائش کریں۔

Few-shot • مثالوں میں ترتیب کی اہمیت: مختلف درجہ بندی کے مسائل میں مثالوں کی ترتیب بدل کر دیں تاکہ ماڈل کی عمومیت میں اضافہ ہو۔

Chain of Thought (CoT) • کے لیے خاص اصول CoT : میں ہمیشہ نتیجہ reason کے بعد دیں اور ٹمپریچر صفر(0) رکھیں۔

• ٹیسٹنگ اور اصلاح: پرامپٹس کی باقاعدہ ٹیسٹنگ کریں اور کارکردگی کی بنیاد پر اصلاح کریں۔

خلاصہ

اس آخری قسط میں ہم نے وائٹ پیپر "Prompt Engineering" سے مؤثر پرامپٹ انجینئرنگ کے لیے بہترین طریقے، تجاویز اور اصول تفصیلی طور پر سمجھے۔ ہم نے سادگی، وضاحت، مثالوں کا استعمال، مختلف فارمیٹس کا مؤثر استعمال اور ہدایات بمقابلہ پابندیوں جیسے اہم اصولوں کا جائزہ لیا۔ اس کے ساتھ ساتھ ماڈلز کی اپ ڈیٹس کے ساتھ پرامپٹس کی ایڈجسٹمنٹ اور وائٹ پیپر بندی کی اہمیت پر بھی بات کی۔

ان تمام معلومات کے ساتھ آپ پرامپٹ انجینئرنگ کے میدان میں کامیابی سے قدم بڑھا سکتے ہیں اور اپنے کام کے معیار کو بہترین سطح تک لے جا سکتے ہیں۔

یہ تحریر اے آئی کی دنیا کے فیس بک پیج پر پوسٹ کی گئی ہے۔

#AI #AIkiDuniya #google #Whitepaper #PromptEngineering